Towards exponential models: Texture synthesis using statistical constraints

Bruno Galerne
bruno.galerne@univ-orleans.fr

Institut Denis Poisson
Université d'Orléans, Université de Tours, CNRS

Master MVA
Cours "Méthodes stochastiques pour l'image"
Lundi 8 mars 2021

Outline

Texture synthesis

Heeger and Bergen

Portilla and Simoncelli

Gatys et al.

Discussion on variational texture synthesis

Outline

Texture synthesis

Heeger and Bergen

Portilla and Simoncelli

Gatys et al.

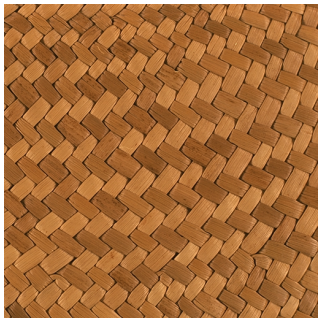
Discussion on variational texture synthesis

What is a texture?

A minimal definition of a **texture** image is an “image containing repeated patterns” [Wei et al., 2009].
The family of patterns reflects a certain amount of randomness, depending on the nature of the texture.
Two main subclasses:

- The ***micro-textures***.

- The ***macro-textures***, constitued of small but discernible objects.



Textures and scale of observation

Depending on the **viewing distance**, the same objects can be perceived either as

- a micro-texture,
- a macro-texture,
- a collection of individual objects.

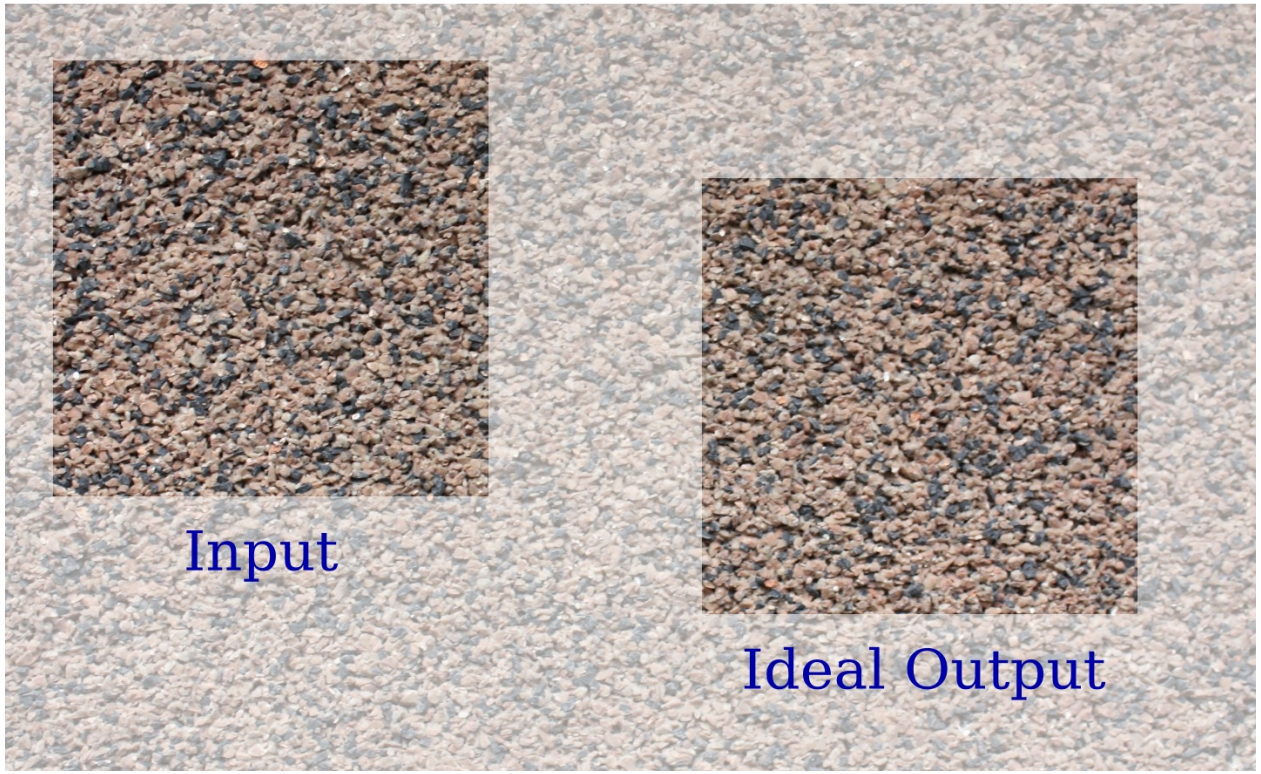
Micro-texture

Macro-texture

Some pebbles

Texture synthesis

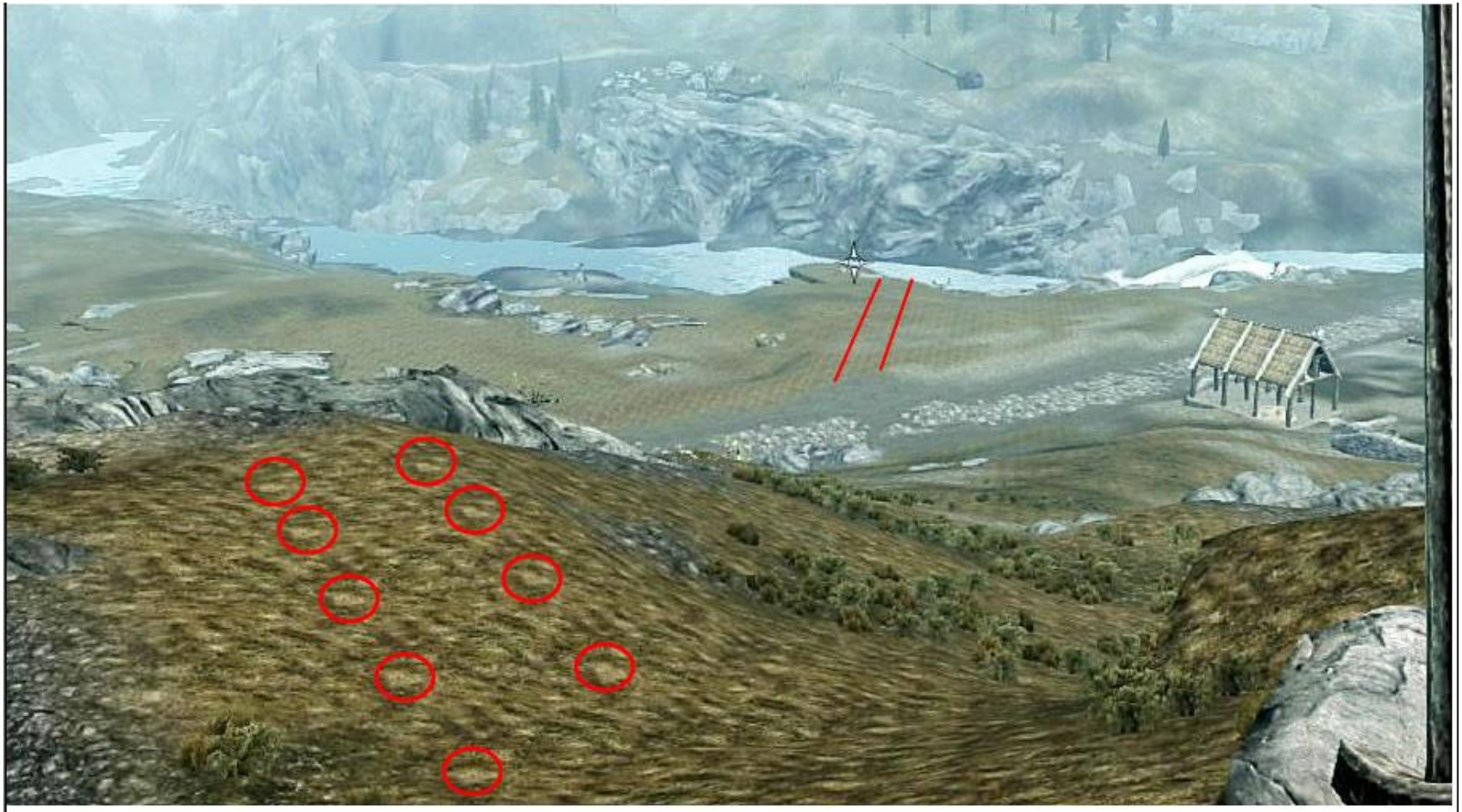
Texture Synthesis: Given an input texture image, produce an output texture image being both visually similar to and pixel-wise different from the input texture.



The output image should ideally be perceived as another part of the same large piece of homogeneous material the input texture is taken from.

Texture synthesis: Motivation

- Important problem in the industry of virtual reality (video games, movies, special effects,. . .).
- Periodic repetition is not satisfying !



2011: Skyrim (Bethesda) *screenshot from Three Parts Theory*

Texture synthesis algorithms

Two main kinds of algorithm:

1. Texture synthesis using statistical constraints:

   Algorithm:

   1.1 Extract some meaningful “statistics” from the input image (e.g. distribution of colors, of Fourier coefficients, of wavelet coefficients,. . .).

   1.2 Compute a “random” output image having the same statistics: start from a white noise and alternatively impose the “statistics” of the input.

   Properties:

   + Perceptually stable
   - Generally not good enough for macro-textures

2. Neighborhood-based synthesis algorithms (or “copy-paste” algorithms):

   Algorithm:

   - Compute sequentially an output texture such that each patch of the output corresponds to a patch of the input texture.
   - Many variations have been proposed: scanning orders, grow pixel by pixel or patch by patch, multiscale synthesis, optimization procedure,. . .

   Properties:

   + Synthesize well macro-textures
   - Can have some speed and stability issue, hard to set parameter...

Texture synthesis algorithms

Two main kinds of algorithm:

1. Texture synthesis using statistical constraints:

   Algorithm:

   1.1 Extract some meaningful “statistics” from the input image (e.g. distribution of colors, of Fourier coefficients, of wavelet coefficients,. . .).

   1.2 Compute a “random” output image having the same statistics: start from a white noise and alternatively impose the “statistics” of the input.

   Properties:

   + Perceptually stable
   - Generally not good enough for macro-textures

2. Neighborhood-based synthesis algorithms (or “copy-paste” algorithms):

   Algorithm:

   - Compute sequentially an output texture such that each patch of the output corresponds to a patch of the input texture.
   - Many variations have been proposed: scanning orders, grow pixel by pixel or patch by patch, multiscale synthesis, optimization procedure,. . .

   Properties:

   + Synthesize well macro-textures
   - Can have some speed and stability issue, hard to set parameter...

Texture synthesis algorithms

Two main kinds of algorithm:

1. Texture synthesis using statistical constraints:

   Algorithm:

   1.1 Extract some meaningful “statistics” from the input image (e.g. distribution of colors, of Fourier coefficients, of wavelet coefficients,. . .).

   1.2 Compute a “random” output image having the same statistics: start from a white noise and alternatively impose the “statistics” of the input.

   Properties:

   + Perceptually stable
   - Generally not good enough for macro-textures

2. Neighborhood-based synthesis algorithms (or “copy-paste” algorithms):

   Algorithm:

   - Compute sequentially an output texture such that each patch of the output corresponds to a patch of the input texture.
   - Many variations have been proposed: scanning orders, grow pixel by pixel or patch by patch, multiscale synthesis, optimization procedure,. . .

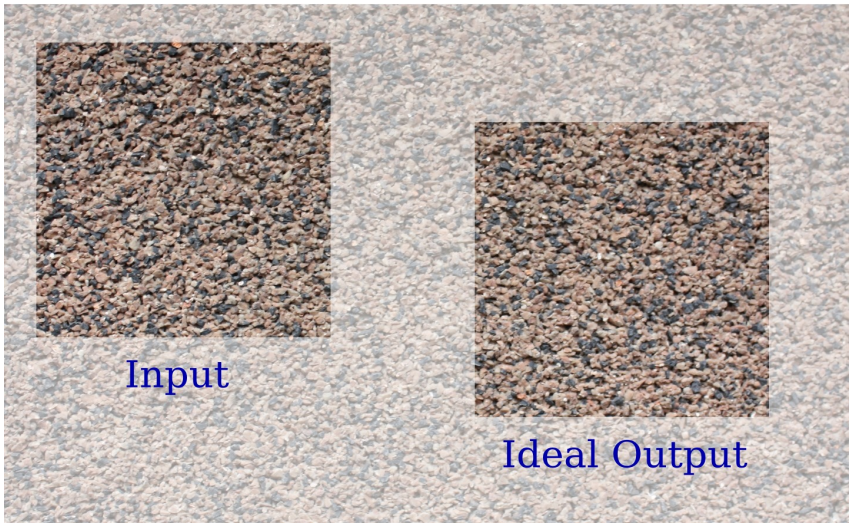
   Properties:

   + Synthesize well macro-textures
   - Can have some speed and stability issue, hard to set parameter...

By-example Texture Synthesis

Notation:

- $\Omega \subset \mathbb{Z}^2$ finite discrete rectangle.
- Image $x : \Omega \to \mathbb{R}^3$
  $x(i) = (x_R(i), x_G(i), x_B(i))$
- π probability distribution on $\mathbb{R}^d$,
  $d = 3|\Omega|$
  (stationary random field).



By-example texture synthesis (with a probabilistic point of view):

- Analysis/Modeling: Estimate a (stationary) distribution π from an exemplar image x_0.
- Synthesis: Sample $x \sim \pi$.

Parametric Texture Synthesis

- Suppose that we have a family of statistical measurements ("features")

$$f = (f_k)_{1 \leqslant k \leqslant p} : \mathbb{R}^d \longrightarrow \mathbb{R}^p$$

  that captures the "perceptual aspect" of the texture, e.g. mean colors, color correlation, etc. (more examples next).
- We want to design a random field X on Ω such that

  $\mathbb{E}[f(X)] = f(x_0)$ **(macrocanonical model: same statistics in average)**.

  or even

  $f(X) = f(x_0)$ a.s. **(microcanonical model: exactly same statistics)**.
- We also need a model which is "as random as possible" (to avoid the trivial solution $X \sim \delta_{x_0}$)
- This will be achieved thanks to the **maximum entropy principle**.

Different Models for Different Statistics

- **Covariance/Fourier Spectrum**
  $\longrightarrow$ Sparse convolution, spectrum painting **[Lewis, 1984]**
  $\longrightarrow$ Spot noise, Random phase noise, Gaussian models
  [Van Wijk, 1991], **[Galerne et al., 2011]**, **[Xia et al., 2014]**
  $\longrightarrow$ Local random phase noise **[Gilet et al., 2014]**

- **Wavelet statistics**
  $\longrightarrow$ Histograms of subbands **[Heeger & Bergen, 1995]**
  $\longrightarrow$ First-order responses to a bank of filters FRAME **[Zhu et al., 1998]**
  $\longrightarrow$ Second-order wavelet statistics **[Portilla & Simoncelli, 2000]**
  $\longrightarrow$ First-order dictionary statistics + spectrum **[Tartavel et al., 2014]**

- **Neural networks statistics**
  $\longrightarrow$ First-order neural statistics **[Lu et al., 2015]**
  $\longrightarrow$ Second-order neural statistics **[Gatys et al., 2015]**

- **Scattering statistics**
  $\longrightarrow$ First-order scattering statistics **[Zhang & Mallat, 2017]**, **[Bruna & Mallat, 2019]**

Different Models for Different Statistics

Green: Macrocanonical Models = maximum entropy model with same statistics in expectation
Red: Microcanonical models = maximum entropy model with exactly same statistics

- **Covariance/Fourier Spectrum**
  $\longrightarrow$ Sparse convolution, spectrum painting **[Lewis, 1984]**
  $\longrightarrow$ Spot noise, **Random phase noise**, **Gaussian models**
  [Van Wijk, 1991], **[Galerne et al., 2011]**, **[Xia et al., 2014]**
  $\longrightarrow$ Local random phase noise **[Gilet et al., 2014]**

- **Wavelet statistics**
  $\longrightarrow$ **Histograms of subbands [Heeger & Bergen, 1995]**
  $\longrightarrow$ **First-order responses to a bank of filters FRAME [Zhu et al., 1998]**
  $\longrightarrow$ **Second-order wavelet statistics [Portilla & Simoncelli, 2000]**
  $\longrightarrow$ **First-order dictionary statistics + spectrum [Tartavel et al., 2014]**

- **Neural networks statistics**
  $\longrightarrow$ **First-order neural statistics [Lu et al., 2015]**
  $\longrightarrow$ **Second-order neural statistics [Gatys et al., 2015]**

- **Scattering statistics**
  $\longrightarrow$ **First-order scattering statistics [Zhang & Mallat, 2017]**, **[Bruna & Mallat, 2019]**

Entropy

Let $\mathcal{P}$ be the set of probability distributions on $\mathbb{R}^d$.

Let μ be a reference probability measure on $\mathbb{R}^d$ (e.g. $\mu(dx) \propto e^{-J(x)}dx$ where $J(x) = \frac{\varepsilon}{2}\|x\|^2$)

The entropy $H : \mathscr{P} \to [-\infty, +\infty)$ (w.r.t. μ) is defined by

$$\forall \pi \in \mathcal{P}, \quad H(\pi) = \begin{cases} -\int_{\mathbb{R}^d} \log\left(\frac{\mathrm{d}\pi}{\mathrm{d}\mu}(x)\right) \frac{\mathrm{d}\pi}{\mathrm{d}\mu}(x)\mu(\mathrm{d}x) & \text{if } \frac{\mathrm{d}\pi}{\mathrm{d}\mu} \text{ exists} \\ -\infty & \text{otherwise} \end{cases}$$

Notice that

- $H(\pi) = -\mathrm{KL}(\pi|\mu)$
- H is strictly concave.

Macrocanonical/Microcanonical Models

Definition

Let $x_0 \in \mathbb{R}^d$ be the exemplar texture and $f : \mathbb{R}^d \to \mathbb{R}^p$ measurable.

- A **microcanonical** model associated with x_0 for the statistics f (with reference measure μ) is a probability distribution $\pi \in \mathcal{P}$ that solves

$$\max H(\pi)$$

  over all $\pi \in \mathcal{P}$ such that $X \sim \pi \;\Rightarrow\; f(X) = f(x_0)$ a.s.

- A **macrocanonical** model associated with x_0 for the statistics f (with reference measure μ) is a probability distribution $\pi \in \mathcal{P}$ that solves

$$\max H(\pi)$$

  over all $\pi \in \mathcal{P}$ such that $\mathbb{E}_{X\sim\pi}[f(X)] = f(x_0)$.

Course plan

Today:

- Variational texture synthesis: Microcanonical models only
- Three algorithms discussed
- Lab session using CNN

Next week:

- Mathematics for macrocanonical models: existence, entropy maximization, exponential models,. . .
- Sampling of macrocanonical models
- Lab session on sampling using Langevin dynamics

Outline

Texture synthesis

Heeger and Bergen

Portilla and Simoncelli

Gatys et al.

Discussion on variational texture synthesis

Portilla and Simoncelli

- Very impressive results.
- Stayed state of the art for statistical texture synthesis for a long time.
- Patch-based method were introduced at the same period.
- Here the textures are synthesized with only 710 parameters.

- Extended to color textures in a Matlab code (no associated publication).
- Recently reproduced by J. Vacher and T. Briand (IPOL preprint) [Vacher and Briand 2020]

Portilla and Simoncelli

- Very impressive results.
- Stayed state of the art for statistical texture synthesis for a long time.
- Patch-based method were introduced at the same period.
- Here the textures are synthesized with only 710 parameters.

- Extended to color textures in a Matlab code (no associated publication).
- Recently reproduced by J. Vacher and T. Briand (IPOL preprint) [Vacher and Briand 2020]

Portilla and Simoncelli: Color results

Outline

Texture synthesis

Heeger and Bergen

Portilla and Simoncelli

Gatys et al.

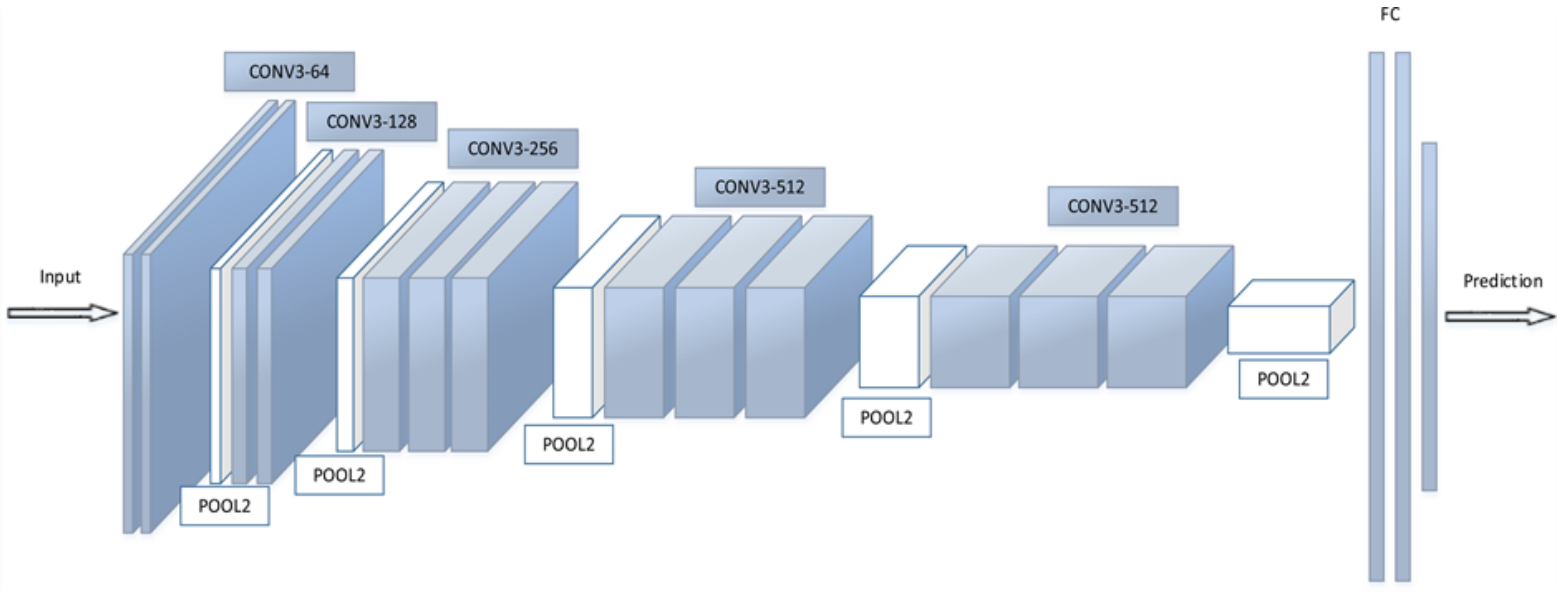
Discussion on variational texture synthesis

Gatys et al algorithm

References: L. Gatys, A. S. Ecker, and M. Bethge, *Texture synthesis using convolutional neural networks*, in Advances in Neural Information Processing Systems, 2015

Convolutional Neural Networks (CNN)

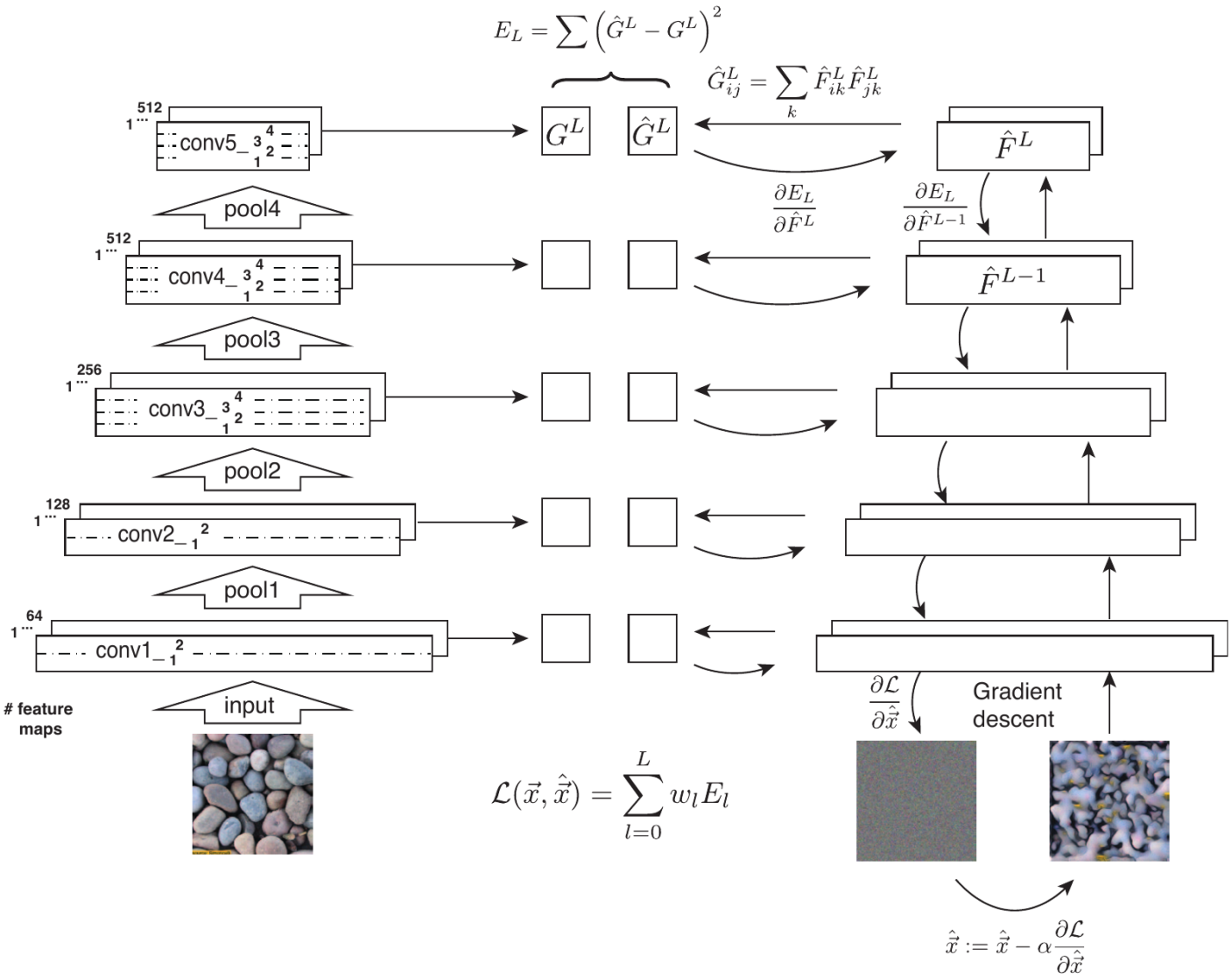
- Main idea: Use the feature layers of a trained deep CNN, namely VGG 19 [Simonyan and Zisserman], as statistics.
- VGG 19 [Simonyan and Zisserman] was trained for image classification.
- It only uses 3×3 convolution kernels followed by RELU (= positive part) and max-pooling.



- For texture analysis, we only use the “pool” layers.
- We do not use the last “fully connected” layers that perform classification.
- VGG 19 is understood as a multiscale nonlinear transform adapted to natural images.

Gatys et al algorithm

- Each feature layer is spatially averaged and the Gram matrix is formed for each layer.
- Texture synthesis consists in minimizing the sum of the Frobenius norm between Gram matrices.



Gatys et al algorithm

Given an example image u and a random initialization x_0, one optimizes the loss function

$$E(x) = \sum_{\text{for selected layers } L} w_L \left\| G^L(x) - G^L(u) \right\|_F^2$$

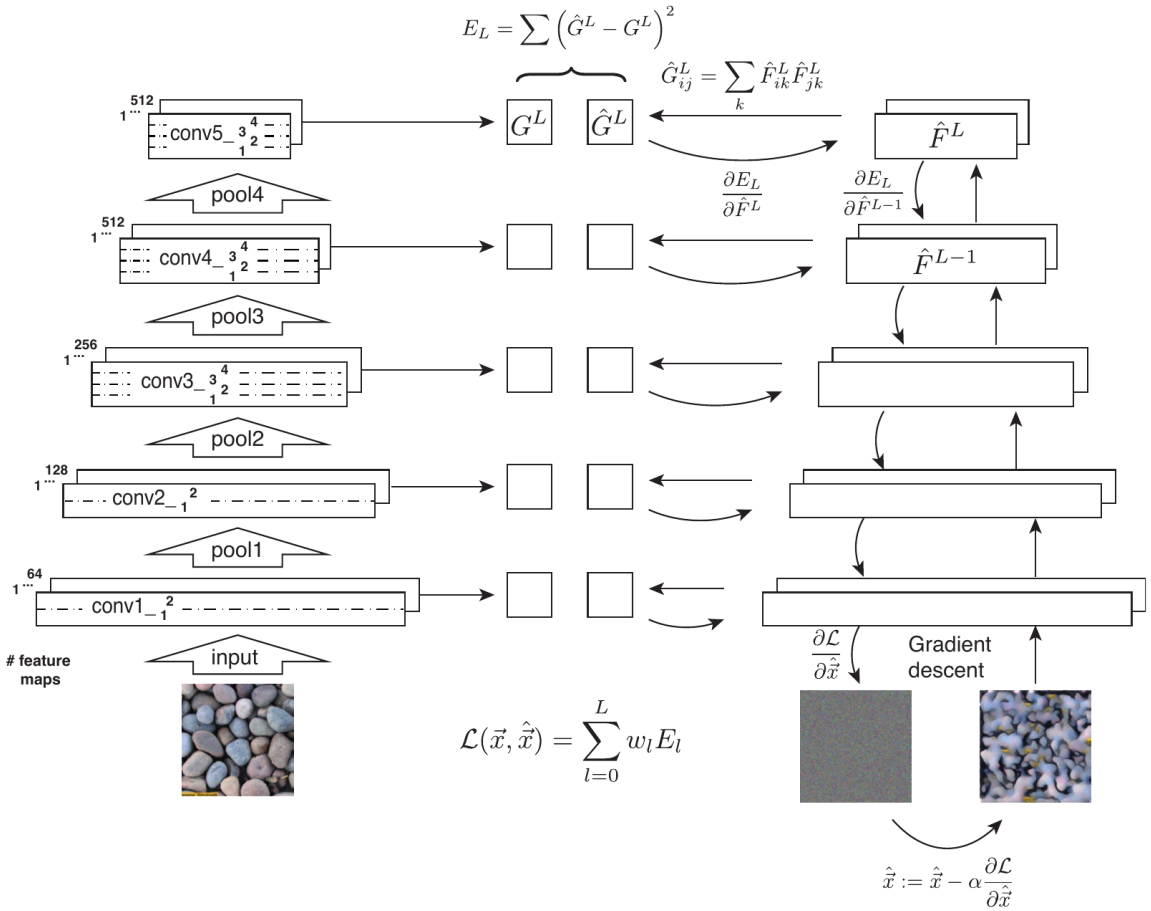
where

- w_L is a weight parameter for each layer
- $\| \cdot \|_F$ is the Frobenius norm
- for an image y and a layer index L, $G^L(y)$ denotes the **Gram matrix** of the VGG-19 features at layer L: if $V^L(y)$ is the feature response of y at layer L that has spatial size $w \times h$ and n channels,

$$G^L(y) = \frac{1}{wh} \sum_{k \in \{0,\ldots,w-1\} \times \{0,\ldots,h-1\}} V^L(y)_k V^L(y)_k^T \in \mathbb{R}^{n \times n}.$$

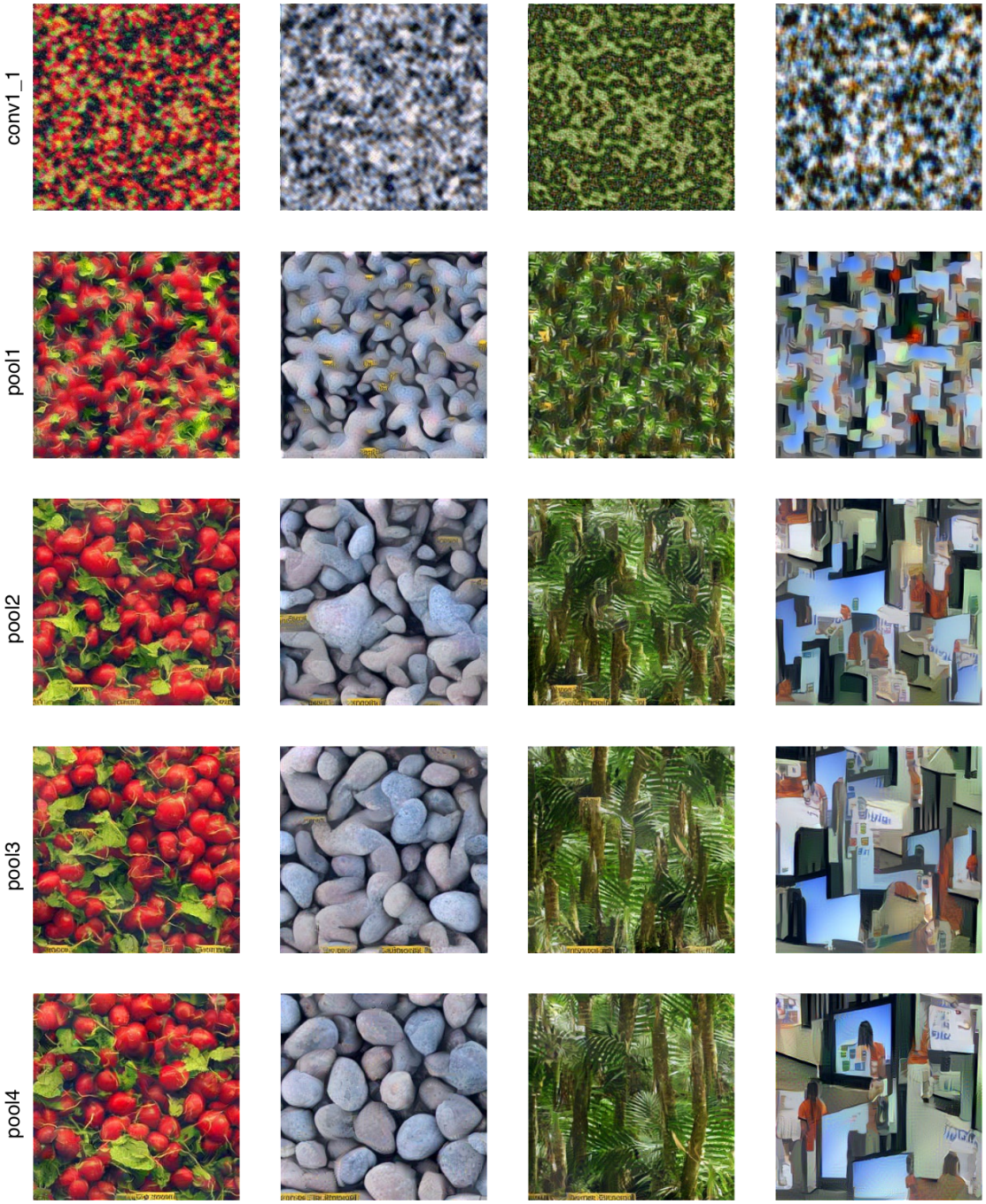
The Gram matrix is a spatial statistics that

Gatys et al algorithm

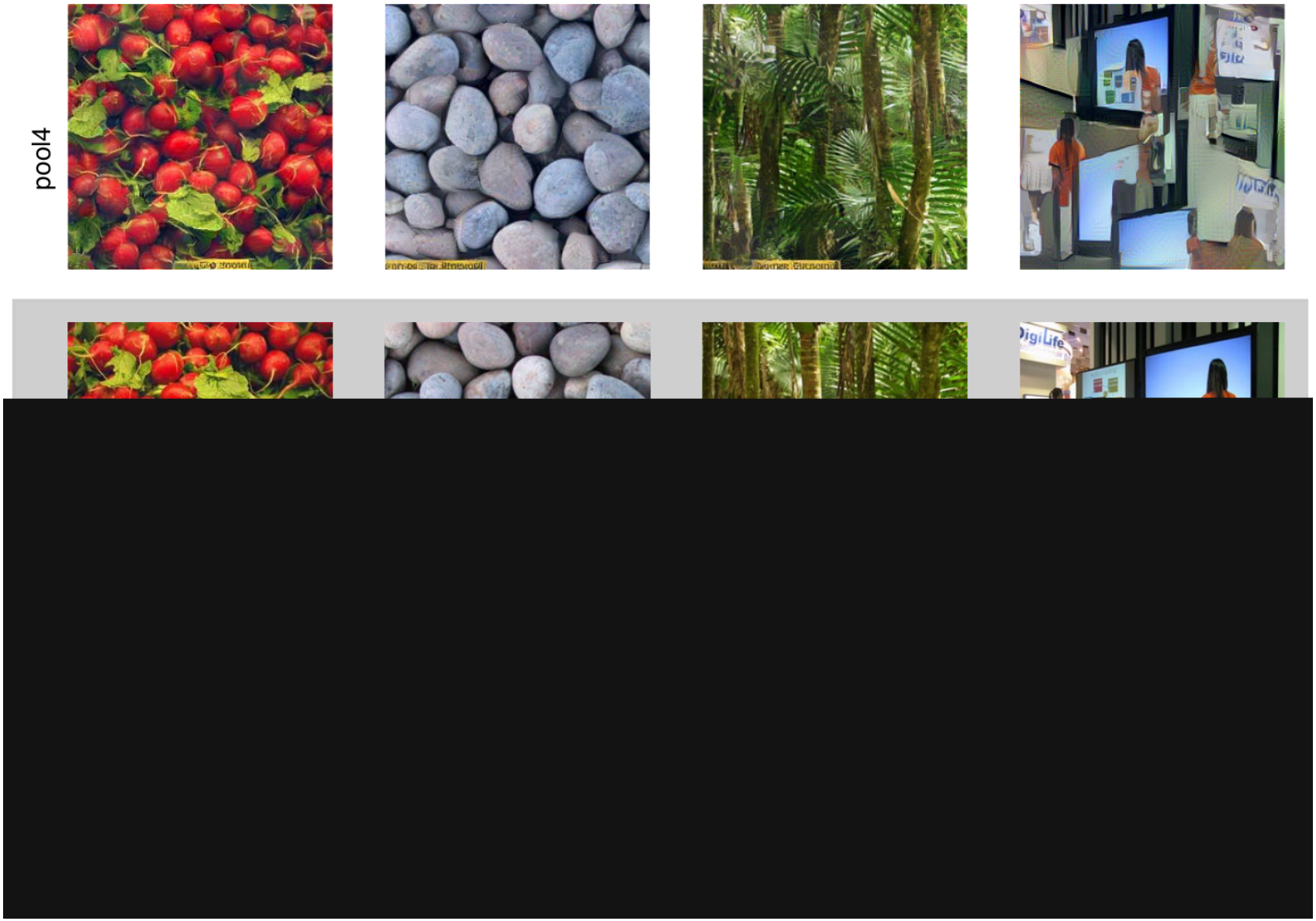


- The gradient of the energy is computed using back-propagation routines.
- The authors use a quasi-Newton algorithm: L-BFGS that stands for Limited-memory Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm.
- It consists in using a low-rank approximation of the Hessian matrix for computing the descent direction.

Gatys et al algorithm: Depth influence



Gatys et al algorithm: Results



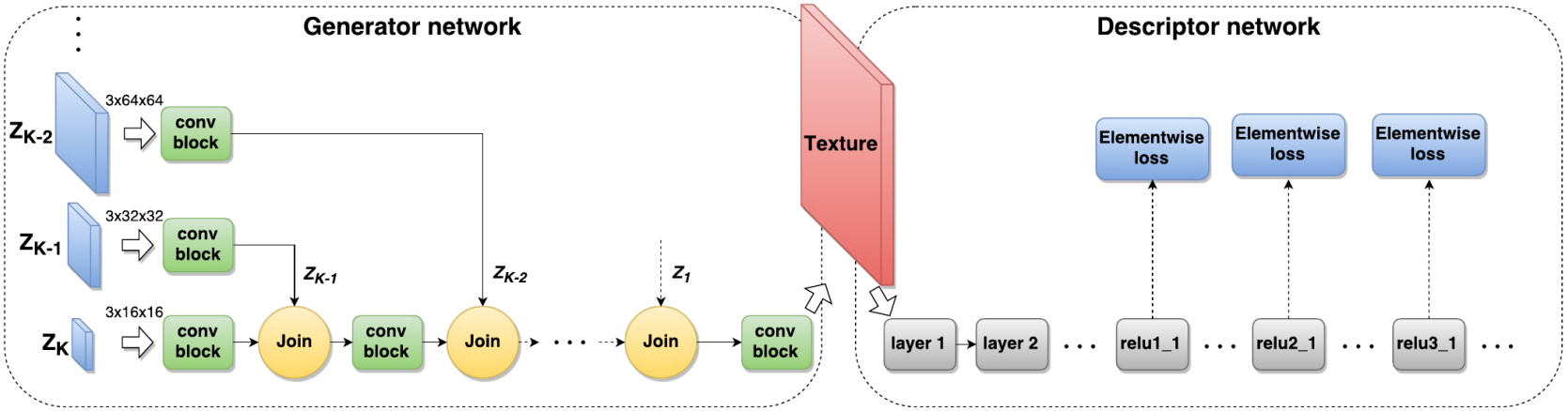
- MORE RESULTS

Gatys et al for texture synthesis and style transfer

- Very nice and clean PyTorch implementation: `https://github.com/leongatys/PytorchNeuralStyleTransfer`
- Today: Practice session based in this code.
- **Very slow** on CPU and computationally demanding with high-end GPU (and memory consuming, e.g. 8 GB of memory for a 1024×1024 image). See practice session.
- Regarding texture modeling, the **number of parameters is huge**: Textures are described by the Gram matrices and the number of elements in the Gram matrices totals 850k. That is 1000 times more than Portilla-Simoncelli !

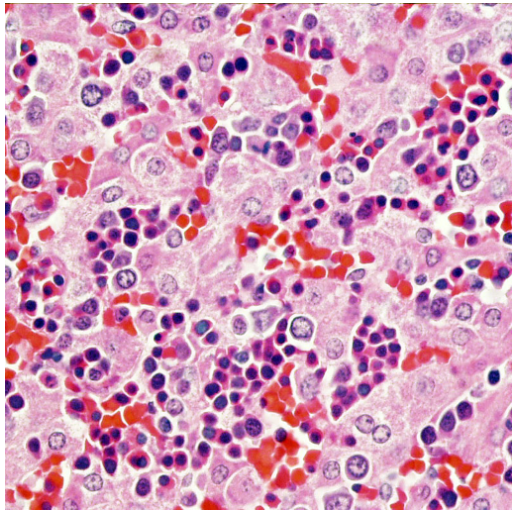
Generative networks for texture synthesis

- A workaround for speeding up synthesis is to train generative forward networks to mimic Gatys algorithm, as proposed by [Ulyanov et al. 2016] (TextureNets).
- Then synthesis is fast but the texture quality is not as good, and a network has to be trained for each new image.

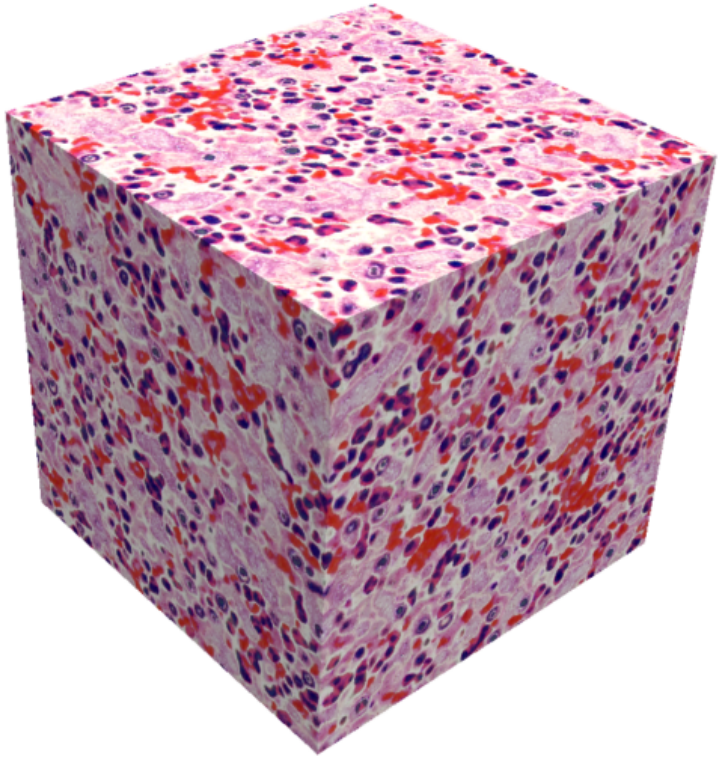


On Demand Solid Texture Synthesis Using Deep 3D Networks

- Recent work with Jorge Gutierrez (PhD), Julien Rabin and Thomas Hurtut [Gutierrez et al 2016].
- Training of generative networks for 3D textures (called solid textures) where the Gatys approach is infeasible.



Input



Output

Generative networks for texture synthesis

- Other recent contributions propose generative networks for universal style transfer/texture synthesis (e.g. [Li et al 2017]).
- This still relies in approximating the Gatys procedure with some approximation for a faster style transfer (eg auto-encoders to invert VGG19 + imposing Gram matrices).

Outline

Texture synthesis

Heeger and Bergen

Portilla and Simoncelli

Gatys et al.

Discussion on variational texture synthesis

Lack of probabilistic modeling

So far the discussed algorithms uses the following methodology.

1. Define statistics of interest $F(x_0)$
2. Define a (non convex) energy E that represents the distance of an image x to the ones of the input x_0, e.g. $E = \|F(x) - F(x_0)\|^2$.
3. Minimize E with some descent algorithm **starting from a white noise image**

Several observations:
Regarding optimization:

- One always have $E(\text{input}) = 0$ that is the input is a global minimum.
- But for texture synthesis we do not want the output to be equal to the input !
- Things are OK in practice since we minimize a non convex energy starting from a white noise that is very far from the input.

Lack of probabilistic modeling

So far the discussed algorithms uses the following methodology.

1. Define statistics of interest $F(x_0)$
2. Define a (non convex) energy E that represents the distance of an image x to the ones of the input x_0, e.g. $E = \|F(x) - F(x_0)\|^2$.
3. Minimize E with some descent algorithm **starting from a white noise image**

Several observations:
Regarding probabilistic modeling:

- There is no randomness except for the initial white noise.
- By minimizing E one tries to correct the excess of disorder of the initial white noise image.
- There is no random field model, but the probable states are the local minima of the energy E, at least the ones one can reach starting from a white noise image... Nothing can be said about the implicit probability distribution, i.e. how the local minima are charged.

Bibliographic references I

- T. Briand, J. Vacher, B. Galerne and J. Rabin, *The Heeger & Bergen Pyramid Based Texture Synthesis Algorithm*, Image Processing On Line, 2014
- L. Gatys, A. S. Ecker, and M. Bethge, *Texture synthesis using convolutional neural networks*, in Advances in Neural Information Processing Systems, 2015
- Gatys, L. A., Ecker, A. S., & Bethge, M. (2016, June). Image style transfer using convolutional neural networks. In Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on (pp. 2414-2423). IEEE.
- J. Gutierrez, B. Galerne, J. Rabin, and T. Hurtut, *On Demand Solid Texture Synthesis using Deep 3D Networks*, Computer Graphics Forum, 2020
- D. J. Heeger and J. R. Bergen, *Pyramid-based texture analysis/synthesis*, SIGGRAPH ’95, 1995
- Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H.Yang, *Universal Style Transfer via Feature Transforms*, Advances in Neural Information Processing Systems 30 (NIPS 2017)

Bibliographic references II

- J.-L. Lisani, A. Buades, and J.-M. Morel. *Image color cube dimensional filtering and visualization*, Image Processing On Line, 2011.
- J. Portilla and E. Simoncelli, *A parametric texture model based on joint statistics of complex wavelet coefficients*, IJCV, 40 (2000)
- E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger. *Shiftable multi-scale transforms*. IEEE Transaction on Information Theory, 1992.
- K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, tech report, 2014
- D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky, *Texture networks: Feed-forward synthesis of textures and stylized images*, in *ICML*, 2016, pp. 1349–1357.
- J. Vacher and T. Briand, *The Portilla-Simoncelli Texture Model: Towards the Understanding of the Early Visual Cortex*, preprint IPOL (with on-line demo)
- Xiang-Yang Wang, Bei-Bei Zhang and Hong-Ying Yang, *Content-based image retrieval by integrating color and texture features*, Multimedia Tools and Applications, 2014

Bibliographic references III

- L.-Y. Wei, S. Lefebvre, V. Kwatra, and G. Turk. *State of the art in example-based texture synthesis*, Eurographics 2009, 2009.
- S. Zhu, Y. Wu, and D. Mumford, *Filters, random fields and maximum entropy (FRAME): Towards a unified theory for texture modeling*, International Journal of Computer Vision, 27 (1998)